

Reduction Rules for Colored Workflow Nets^{*}

Javier Esparza and Philipp Hoffmann

Technische Universität München

Abstract. We study Colored Workflow nets [8], a model based on Workflow nets [14] enriched with data. Based on earlier work by Esparza and Desel on the negotiation model of concurrency [3,4], we present reduction rules for our model. Contrary to previous work, our rules preserve not only soundness, but also the data flow semantics. For free choice nets, the rules reduce all sound nets (and only them) to a net with one single transition and the same data flow semantics. We give an explicit algorithm that requires only a polynomial number of rule applications.

1 Introduction

Workflow Petri nets [14,13] are a very successful formalism for modeling and analyzing business processes. They have become the most popular formal backend for graphical notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), or UML Activity Diagrams, which typically do not have a formal semantics. By translating the basic constructs of such languages into Petri nets one gets access to a large variety of analysis techniques and tools.

One of these analysis techniques is *reduction*. Reduction algorithms are a very efficient analysis technique for workflows, EPCs, AND-XOR graphs and other models (see for instance [11,15,18,21]). They consist of a set of *reduction rules*, whose application allows one to simplify the workflow while preserving important properties. Reduction aims to elude the state-explosion problem, and, when the property does not hold, provides error diagnostics in the form of an irreducible graph [15]. Moreover, for certain classes of nets the rules can be *complete*, meaning that they reduce all workflows satisfying the property to some unique canonical workflow (and only them); in this case, reduction provides a decision algorithm for the property that avoids any kind of state-space exploration. Reduction algorithms are an important part of the well-known Woflan tool [20,9].

Free choice workflow nets (also called workflow graphs) are a class of workflow nets that captures many control-flow constructs of BPMN, EPC, or Activity Diagrams (see [14], or [6] for a very recent study). In [15] it is shown that a certain set of reduction rules for free choice workflow models, originally presented in [2], preserves the *soundness* property, and is complete. Soundness is a fundamental analysis problem for workflows [14,16]. Loosely speaking, a workflow net is sound

^{*} This work was partially funded by the DFG Graduiertenkolleg 1480 (PUMA).

if a distinguished marking signaling successful termination is reachable from any reachable marking. The reduction algorithm provides a polynomial-time decision procedure for soundness, in sharp contrast with the fact that deciding soundness is at least PSPACE-hard for general workflow nets¹.

However, the rules of [2] have two important shortcomings. First, while they preserve soundness, they do not preserve any property concerning *data*. Workflows manipulating data can be modeled as *colored* workflow nets [8], where tokens carry data values, and transitions transform a tuple of values for its input places into a tuple of values for its output places. The *linearly dependent place rule* (Rule 2 in Chapter 7 of [2]) allows one to *remove* place p from a net, if it is redundant in the sense that there are other places which together have the same incoming and outgoing transitions as p . However, this reduction does not make sense for the colored workflow net: the tokens on p might hold a value needed by an outgoing transition t to compute the value of the produced tokens! Loosely speaking, the application of the rule destroys the dataflow semantics of the net.

The second shortcoming is that the linearly dependent place rule is not correct for arbitrary workflow nets, only for free choice ones ([2], page 145²). Since not all industrial business processes are free choice (30% of our benchmarks in Section 5 are non-free choice), this considerably reduces the applicability of the rules.

The most satisfactory solution to these two problems would be to replace the linearly dependent place rule by rules extensible to colored nets, while keeping completeness. However, this problem has remained open for over 15 years.

In this paper we solve this problem and present a set of surprisingly simple rules that overcomes the shortcomings. First, the rules can be applied to arbitrary colored workflow nets. Second, they preserve not only the sound/unsound character of the net, but also the *input/output relation* of the workflow; more precisely, the original workflow net has a firing sequence that transforms an entry token with value v_{in} into an exit token with value v_{out} iff the net after the reduction also has such a sequence. Therefore, the rules can be applied to decide any property of the input/output relation. Finally, the new rules are complete for free choice workflow nets.

Our results rely on previous work on *negotiations*, a model of concurrency introduced in [3,4]. Negotiations share many features with Petri nets, but, unlike Petri nets, are a structured model of communicating sequential agents. In [4] a complete set of reduction rules for the class of *deterministic negotiations* is presented. We generalize the results of [4] to show that a similar set of rules is correct for arbitrary workflow nets, and complete for free choice workflow nets. Since the proofs of [4] make strong use of the agent structure, we must substantially modify them, and in fact write many of them from scratch. Moreover, because

¹ The exact complexity depends on the specifics of the workflow model, for instance whether the workflow Petri net is assumed to be 1-safe or not.

² The example of page 145 is not a workflow net, but can be easily transformed into one.

of the agent structure of negotiations, workflow nets obtained as translations of negotiations are automatically 1-safe. Therefore, the results cannot be used to deal with variants of the soundness notion, like k -soundness or generalized soundness [16]. Making use of the theory of free choice nets we can however show that our rules are still correct and complete for these variants.

Finally, and as a third contribution of the paper, we report on some experimental results. In [4] only the rules and the completeness result are presented, but neither a specific algorithm prescribing a concrete strategy to decide which rule to apply at which point, nor an implementation and experimental validation. In this paper we report on a prototype implementation, and on experimental results on a benchmark suite of nearly 2000 workflows derived from industrial business processes.

Other related work. The soundness problem has been extensively studied, both from a theoretical and a practical point of view, and very efficient verification algorithms have been developed (see e.g. [16] for a comprehensive survey). Our approach is not more efficient for checking soundness than the ones of e.g. [5], but can also be applied to checking arbitrary properties of the input/output relation, while retaining completeness. In [10,12] state-space exploration of workflows is performed to identify data flow anti-patterns (like a variable being assigned a value during an execution, but never being read afterwards). Our technique aims at avoiding state-space exploration and considers properties of the input/output relation.

The paper is organized as follows. Section 2 defines workflow nets, free choice nets, and soundness. Section 3 presents our reduction rules and proves them correct. In Section 4 we first show completeness for acyclic nets and then extend the result to cyclic nets. Section 5 presents experimental results on the benchmarks of [17,5]. Finally, Section 6 contains some conclusions and open questions. The proofs of all results can be found in the appendix.

2 Workflow Nets and Colored Workflow Nets

We recall the definitions of workflow nets and the soundness property.

Definition 1 (Workflow net). [14] *A Workflow net (WF net) is a quintuple (P, T, F, i, o) where*

- P is a finite set of places.
- T is a finite set of transitions ($P \cap T = \emptyset$).
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.
- $i, o \in P$ are places such that i has no incoming arcs, o has no outgoing arcs.
- The graph $(P \cup T, F \cup (o, i))$ is strongly connected.

We write $\bullet p$ and p^\bullet to denote the input and output transitions of a place p , respectively, and similarly $\bullet t$ and t^\bullet for the input and output places of a transition t . A marking M is a function from P to the natural numbers that

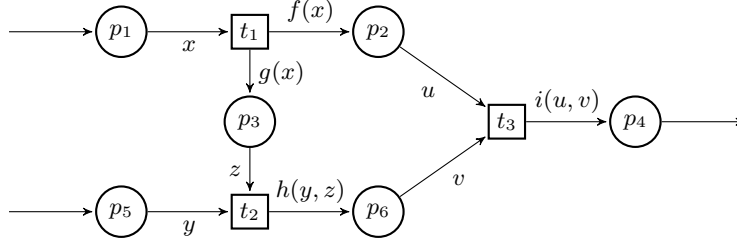


Fig. 1. A partial workflow net with data

assigns a number of tokens to each place. A transition t is enabled at M if all places of $\bullet t$ contain at least one token in M . An enabled transition may fire, removing a token from each place of $\bullet t$ and adding one token to each place of $t\bullet$. The *initial marking* (*final marking*) of a workflow net puts one token on place i (on place o), and no tokens elsewhere. A marking is *reachable* if some sequence of transition firings leads from the initial marking to it. We call elements in $P \cup T$ the nodes of the workflow net.

Definition 2 (Soundness). [14] A WF net $\mathcal{W} = (P, T, F, i, o)$ is sound if

- the final marking is reachable from any reachable marking, and
- every transition occurs in some firing sequence starting from the initial marking.

When modeling a workflow, it is useful to model not only control flow but also data flow. We do so by means of Colored Workflow nets.

Definition 3 (Colored WF net). [8] A colored WF net (CWF net) is a tuple $\mathcal{W} = (P, T, F, i, o, V, \lambda)$ where (P, T, F, i, o) is a WF net, V is a function that assigns to every place $p \in P$ a color set C_p and λ is a function that assigns to each transition $t \in T$ a left-total relation $\lambda(t) \subseteq \prod_{p \in \bullet t} C_p \times \prod_{p \in t\bullet} C_p$ between the values of the input places and those of the output places of t .

A colored marking M of \mathcal{W} is a function that assigns to each place p a multiset $M(p)$ over C_p , interpreted as a multiset of colored tokens currently on p . A colored marking is *initial* (*final*) if it puts one token on place i (on place o), of any color in C_i (C_o), and no tokens elsewhere.

Observe that there are as many initial markings as elements in C_i . To distinguish between input and output values of a transformer λ , we separate them by a \rightarrow .

Consider the partial workflow net in Figure 1 and take $C_p = \mathbb{N}$ for every place p of the net. An example of a colored marking could be the marking $(\{3\}, \emptyset, \emptyset, \emptyset, \{2, 4\}, \emptyset)$ which puts a token of color 3 on p_1 and two tokens, one of color 2 and one of color 4, on p_5 . If $f(x) = x + 1$ and $g(x) = x + 2$, then we have $\lambda(t_1) = \{(n \rightarrow n + 1, n + 2) \mid n \geq 0\}$.

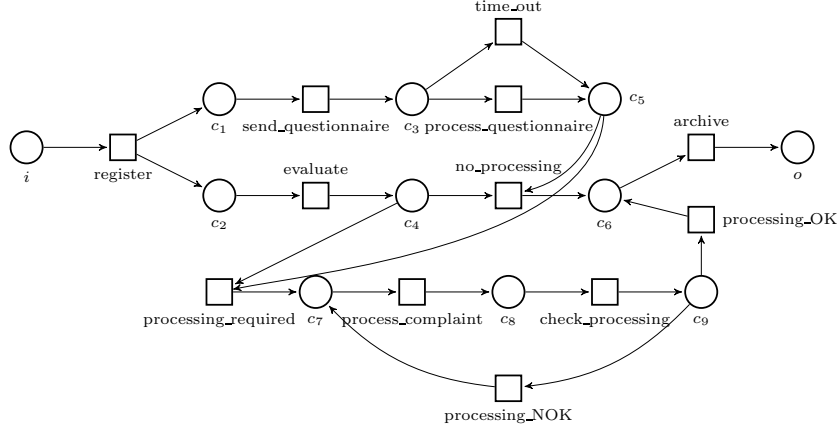


Fig. 2. Insurance claim process

We call $\lambda(t)$ the transformer associated with t . When a transition t fires, the colored marking changes in the expected way [8]: (a) remove a token from each input place of t ; (b) choose an element of $\lambda(t)$ whose projection onto the input places matches the tuple of removed tokens; (c) add the projection of $\lambda(t)$ onto the output places to the output places of t . We write $M \xrightarrow{t} M'$ to denote that t is enabled at M and its firing leads to M' . For example, the colored marking $(\{3\}, \emptyset, \emptyset, \emptyset, \{2, 4\}, \emptyset)$ enables transition t_1 , and taking $h(y, z) = y \cdot z$ we have

$$(\{3\}, \emptyset, \emptyset, \emptyset, \{2, 4\}, \emptyset) \xrightarrow{t_1} (\emptyset, \{4\}, \{5\}, \emptyset, \{2, 4\}, \emptyset) \xrightarrow{t_2} (\emptyset, \{4\}, \emptyset, \emptyset, \{4\}, \{10\}) .$$

2.1 A colored version of the insurance claim example

We extend the well known insurance complaint process of [14] with data. The workflow is shown in Figure 2. After initial registration of the complaint, a questionnaire is sent to the complainant. In parallel, the complaint is evaluated. The evaluation decides whether processing is required. In that case, the processing takes place (e.g. by some employee) and is checked for correctness (e.g. by a senior employee) which may either lead to another round of processing if an error is found, or the processing ends. Finally, the complaint is archived.

We add colors to keep track of the status of the complaint and its estimated cost for the company, modeled by a number in the interval $[1..10]$ (see Table 1). Furthermore each claimant belongs to a customer group, either A or B. A's and B's insurance policies entitle them, respectively, to the full cost or to half the cost of the damage. The color sets of places i, o, c_2, c_6 are the pairs $\{A, B\} \times [1..10]$, modeling the customer group and the cost of the claim as estimated by the customer. The colors of place c_4 additionally contain the result of the evaluation: PR (process) or NPR (do not process). Colors of c_5 store the result of the questionnaire: the answer to the question “was it your fault?” (YES/NO), or a

$$\begin{aligned}
C_i = C_o = C_{c_2} = C_{c_6} &= \{A, B\} \times [1..10] & C_{c_7} &= C_i \times C_{c_5} \\
C_{c_1} = C_{c_3} &= \{\bullet\} & C_{c_8} &= C_{c_7} \times [1..10] \\
C_{c_4} &= C_i \times \{\text{PR}, \text{NPR}\} & C_{c_9} &= C_{c_7} \times ([1..10] \cup \{\text{ERR}\}) \\
C_{c_5} &= \{\text{YES}, \text{NO}, \text{TO}\}
\end{aligned}$$

$$\begin{aligned}
\lambda(\text{register}) &= \{(x, k \rightarrow \{\bullet\}) \times \{x, k\} \mid 1 \leq k \leq 10\} \\
\lambda(\text{send_questionnaire}) &= \{(\bullet \rightarrow \bullet)\} \\
\lambda(\text{time_out}) &= \{(\bullet \rightarrow \text{TO})\} \\
\lambda(\text{process_questionnaire}) &= \{(\bullet \rightarrow \text{YES}), (\bullet \rightarrow \text{NO})\} \\
\lambda(\text{evaluate}) &= \{(x, k \rightarrow x, k, \text{NPR}) \mid 1 \leq k \leq 3\} \\
&\quad \cup \{(x, k \rightarrow x, k, \text{PR}) \mid 4 \leq k \leq 10\} \\
\lambda(\text{no_processing}) &= \{(x, k, \text{NPR}, q \rightarrow x, k) \mid 1 \leq k \leq 3\} \\
\lambda(\text{processing_required}) &= \{(x, k, \text{PR}, q \rightarrow x, k, q) \mid 4 \leq k \leq 10\} \\
\lambda(\text{process_complaint}) &= \{(x, k, q \rightarrow x, k, q, v) \mid 4 \leq k \leq 10, 1 \leq v \leq k\} \\
\lambda(\text{check_processing}) &= \{(x, k, v, q \rightarrow x, k, q, v) \mid x = A, 4 \leq k \leq 10, v = k\} \\
&\quad \cup \{(x, k, v, q \rightarrow x, k, q, v) \mid x = B, 4 \leq k \leq 10, v = k/2\} \\
&\quad \cup \{(x, k, v, q \rightarrow x, k, q, \text{ERR}) \mid \text{otherwise}\} \\
\lambda(\text{processing_NOK}) &= \{(x, k, q, \text{ERR} \rightarrow x, k, q) \mid 4 \leq k \leq 10\} \\
\lambda(\text{processing_OK}) &= \{(x, k, q, v \rightarrow x, v) \mid 4 \leq k \leq 10, 1 \leq v \leq 10\} \\
\lambda(\text{archive}) &= \{(x, v \rightarrow x, v) \mid (x, v) \in C_{c_6}\}
\end{aligned}$$

Table 1. Color sets and transformers for the insurance claim workflow

time out (TO). In place c_7 , the information from c_4 and c_5 is put together, and in c_8 the result of the first processing is added. Finally, tokens in c_9 can have the same values as those in c_8 , plus an additional value ERR if the check at transition **check_processing** reveals a miscalculation. Tokens in c_6 and o store the amount that was actually paid by the company after the processing was successful (or without processing).

Assume that the company's policy is to accept all claims which are evaluated to a value of 3 or less without any further processing, and process all other claims. The transformers modeling this policy are given in Table 1, where $x \in \{A, B\}$ and $q \in \{\text{YES}, \text{NO}, \text{TO}\}$ unless otherwise stated. Division by 2 is assumed to be integer division.

All transformers are self-explanatory except perhaps **process_complaint** and **check_processing**. In **process_complaint**, an employee may lower the customer's estimate k to a new value v . In **check_processing**, a senior employee checks that the employee made no mistake (modeled by the fact that v must be $k/2$ or k depending on the customer group). If the check fails, an error flag is set and the processing is repeated.

Apart from the soundness of the workflow, we wish to check the following property: if two customers in the same group register insurance complaints, then the one claiming a higher amount also receives a higher amount (notice that our ideal insurance company does not reject any complaint). We shall use our reduction algorithm to check that the property holds for customers of group A, but not for customers of group B.

The attentive reader may have noticed that the semantics of colored nets allows, e.g., to take the transition `no_processing` even when the evaluation indicates that processing is necessary. This can easily be dealt with by introducing additional error values that are then propagated until the end. We omit them to ease the reading and assume that `no_processing` and `processing` are taken according to the result of `evaluate`, and similarly in other cases.

2.2 Summaries and Equivalence

Since a workflow net describes a process starting at i and ending at o , it is interesting to study the input/output relation or *summary* of the whole process.

Definition 4 (Summary and equivalence). *Let \mathcal{W} be a colored WF net. Let \mathcal{M}_i and \mathcal{M}_o be the sets of initial and final colored markings of \mathcal{W} . The summary of \mathcal{W} is the relation $S \subseteq \mathcal{M}_i \times \mathcal{M}_o$ given by: $(M_i, M_o) \in S$ iff M_o is reachable from M_i . Two colored WF nets are equivalent iff they are both sound or both unsound, and have the same summary.*

Our rules aim to reduce CWF nets while preserving equivalence. If we are able to reduce a CWF to another one with one single transition t , then the summary is given by $\lambda(t)$, and we say that the CWF has been *completely reduced* and we have *computed the summary*. Since this CWF net is obviously sound and rules preserve equivalence, if a CWF net can be completely reduced, then it is sound. We prove that our rules preserve equivalence for all CWF nets, and give an algorithm that completely reduces all sound *free choice CWF nets*, defined below, by means of a polynomial number of rule applications.

In Section 4 we compute the summary of the free choice CWF net of Figure 2 using our reduction procedure. The result (where we write $M_i \Rightarrow M_o$ instead of $(M_i, M_o) \in S$, and omit the error values) is:

$$\{(A, k \Rightarrow A, k) \mid 1 \leq k \leq 10\} \cup \{(B, k \Rightarrow B, k) \mid 1 \leq k \leq 3\} \\ \cup \{(B, k \Rightarrow B, k/2) \mid 4 \leq k \leq 10\}$$

Since the summary contains $(B, 3 \Rightarrow B, 3)$ and $(B, 4 \Rightarrow B, 2)$, the company policy does not satisfy the desired property for customers of group B.

2.3 Free choice Workflow Nets

We recall the definition of free choice workflow nets [2,14].

Definition 5 (Free choice workflow nets). *A workflow net $\mathcal{W} = (P, T, F, i, o)$ is free choice (FC) if for every two places $p_1, p_2 \in P$ either $p_1^\bullet \cap p_2^\bullet = \emptyset$ or $p_1^\bullet = p_2^\bullet$.*

The net of Figure 2 is free choice. We also need to introduce clusters, and the new notion of free choice cluster and free choice node.

Definition 6 (Clusters, free choice nodes). [2] Let $\mathcal{W} = (P, T, F, i, o)$ be a workflow net. The cluster of $x \in P \cup T$ is the unique smallest set $[x] \subseteq P \cup T$ satisfying: $x \in [x]$, if $p \in P \cap [x]$ then $p^\bullet \subseteq [x]$, and if $t \in T \cap [x]$, then ${}^\bullet t \subseteq [x]$. A set $X \subseteq P \cup T$ is a cluster if $X = [x]$ for some x . A cluster c is free choice if $(p, t) \in F$ for every $p \in P \cap c$ and $t \in T \cap c$. A node x is free choice if $[x]$ is a free choice cluster.

The sets $\{c_3\} \cup c_3^\bullet$ and $\{c_4, c_5\} \cup c_4^\bullet \cup c_5^\bullet$ are free choice clusters of the net of Figure 2. It is easy to see that clusters are equal or disjoint, and therefore the clusters of \mathcal{W} are a partition of $P \cup T$. Further, we have $[i] \cap P = \{i\}$ and $[o] = \{o\}$. Finally, we have that \mathcal{W} is free choice iff all its nodes are free choice.

We say that a marking M marks a cluster c if it marks *all* places in c . Observe that if a cluster is marked, then all its transitions are enabled. We say that a cluster fires if one of its transitions fires.

3 Reduction rules

We present a set of three *reduction rules* for CWF nets similar to those used for transforming finite automata into regular expressions [7].

A reduction rule, or just rule, is a binary relation on the set of CWF nets. For a rule R , we write $W_1 \xrightarrow{R} W_2$ for $(W_1, W_2) \in R$. A rule R is *correct* if it preserves equivalence, i.e., if $W_1 \xrightarrow{R} W_2$ implies that W_1 and W_2 are equivalent.

Given a set of rules $\mathcal{R} = \{R_1, \dots, R_k\}$, we denote by \mathcal{R}^* the transitive closure of $R_1 \cup \dots \cup R_k$. We say that \mathcal{R} is *complete* for a class of CWF nets if for every sound CWF net \mathcal{W} in that class there is a CFW net \mathcal{W}' consisting of a single transition between the two only places i and o such that $\mathcal{W} \xrightarrow{\mathcal{R}^*} \mathcal{W}'$.

We describe rules as pairs of a *guard* and an *action*. $W_1 \xrightarrow{R} W_2$ holds if W_1 satisfies the guard, and W_2 is a possible result of applying the action to W_1 .

Merge rule. Intuitively, the *merge rule* merges two transitions with the same input and output places into one single transition.

Definition 7. Merge rule

Guard: \mathcal{W} contains two distinct transitions $t_1, t_2 \in T$ such that ${}^\bullet t_1 = {}^\bullet t_2$ and $t_1^\bullet = t_2^\bullet$.

Action: (1) $T := (T \setminus \{t_1, t_2\}) \cup \{t_m\}$, where t_m is a fresh name.

(2) $t_m^\bullet := t_1^\bullet$ and ${}^\bullet t_m := {}^\bullet t_1$.

(3) $\lambda(t_m) := \lambda(t_1) \cup \lambda(t_2)$.

Iteration rule. Loosely speaking, the iteration rule replaces arbitrary iterations of a transition by a single transition with the same effect.

Definition 8. *Iteration rule*

Guard: \mathcal{W} contains a free choice cluster c with a transition $t \in c$ such that $t^\bullet = \bullet t$.

Action: (1) $T := (T \setminus \{t\})$.

(2) For all $t' \in c \setminus \{t\}$: $\lambda(t') := \lambda(t)^* \cdot \lambda(t')$ where $\lambda(t)^* = \sum_{i \geq 0} \lambda(t)^i$, and $\lambda(t)^0$ is the identity relation.

Observe that $\lambda(t)^*$ captures the fact that t can be executed arbitrarily often.

Shortcut rule. The shortcut rule merges transitions of two clusters, one of which will occur as a consequence of the other, into one single transition with the same effect.

Definition 9. A transition t unconditionally enables a cluster c if $c \cap P \subseteq t^\bullet$.

Observe that if t unconditionally enables c and a marking M enables t , then the marking M' given by $M \xrightarrow{t} M'$ enables every transition in c .

Definition 10. *Shortcut rule*

Guard: \mathcal{W} contains a transition t and a free choice cluster $c \notin \{[o], [t]\}$ such that t unconditionally enables c .

Action: (1) $T := (T \setminus \{t\}) \cup \{t'_s \mid t' \in c\}$, where t'_s are fresh names.

(2) For all $t' \in c$: $\bullet t'_s := \bullet t$ and $t'_s \bullet := (t^\bullet \setminus \bullet t') \cup t' \bullet$.

(3) For all $t' \in c$: $\lambda(t'_s) := \lambda(t) \cdot \lambda(t')$.

(4) If $\bullet p = \emptyset$ for all $p \in c$, then remove c from \mathcal{W} .

We also use a restricted version of this rule, called the *d-shortcut rule*. This rule is obtained by adding an additional guard to the shortcut rule: $|c \cap T| = 1$. This guard guarantees that the number of edges does not increase when the d-shortcut rule is applied.

Figure 3 shows a sequence of reductions illustrating the definitions of the rules. Notice that the graphical description does not contain the transformer information. A second example of reduction in which the workflow net also exhibits concurrency is shown in Section 4.1.

Theorem 1. *The merge, shortcut and iteration rules are correct for CWF nets.*

4 Reduction Procedure

We show that the rules presented in the previous section summarize all sound FC-CWF nets in polynomial time. The proof is very involved, and we can only sketch it.

We first show that acyclic FC-CWF nets can be completely reduced.

Definition 11 (Graph). *The graph of a CWF net is the graph $(P \cup T, F)$. A CWF net is acyclic if its graph is acyclic.*

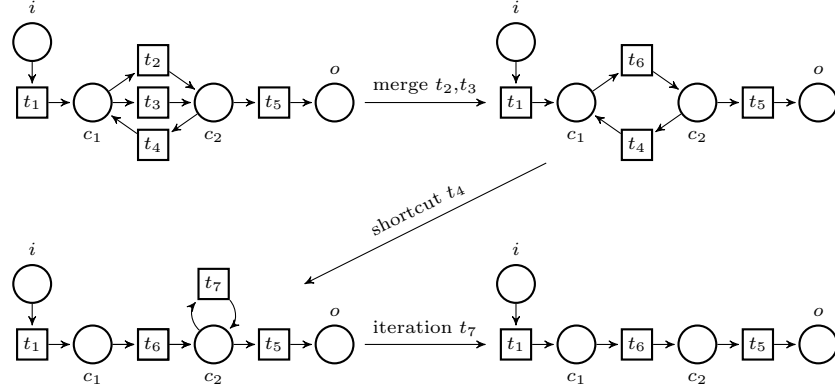


Fig. 3. Example of rule applications

Theorem 2. *The merge and d-shortcut rule are complete for acyclic FC-CWF nets.*

In the cyclic case we need the notion of *synchronizer of a loop*. Although a similar concept was already used in [4], the definition there exploits the fact that negotiations are a structured model of communicating sequential agents. Since workflow nets do not have such a structure, we need a different definition.

Definition 12 (Loop). *Let \mathcal{W} be a CWF net. A non-empty transition sequence σ is a loop of \mathcal{W} if $M \xrightarrow{\sigma} M$ for some reachable marking M .*

Definition 13 (Synchronizer). *Let \mathcal{W} be a WF net. A free choice transition t synchronizes a loop σ if t appears in σ and for every reachable marking M : if M enables t , then $M(p) = 0$ for every $p \in (\bigcup_{t' \in \sigma, t' \neq t} \bullet t')$. A free choice transition is a synchronizer if it synchronizes some loop.*

Consider the insurance claim net, replacing the part between the places c_7 and c_9 by Figure 4. The sequence **process check1 check2 combine processing_NOK** is a loop. Transitions **process**, **combine**, and **processing_NOK** are synchronizers, but **check1** and **check2** are not. We use synchronizers to define fragments of \mathcal{W} on which to apply our rules.

Definition 14 (Fragment). *Let \mathcal{W} be a CWF net and let t be a synchronizer of \mathcal{W} . The fragment \mathcal{W}_t contains all transitions appearing in all loops synchronized by t , together with their input and output places, and the arcs connecting them.*

In our example, the fragment $\mathcal{W}_{\text{process}}$ is exactly the net of Figure 4. Our procedure selects a synchronizer t and applies the rules to \mathcal{W}_t until, loosely speaking, all loops synchronized by t are removed from the net, and t is no longer a synchronizer. The next lemma shows that when no synchronizers can be found anymore, the workflow net is acyclic, and so can be completely reduced by Theorem 2.

Lemma 1. *Every sound cyclic FC-CWF net has at least one synchronizer.*

Proof Sketch. We first show that in every sound cyclic FC-CWF net there exists a loop. We then inspect minimal loops and show that they must include a synchronizer. The proof constructs a transition sequence that pushes one token towards the final marking while all other tokens stay inside the loop. Should no synchronizer be present in the loop, this sequence ends in a dead lock contradicting soundness.

Given two synchronizers t and t' , we say $\mathcal{W}_t \preceq \mathcal{W}_{t'}$ if every node of \mathcal{W}_t is also a node of $\mathcal{W}_{t'}$. The relation \preceq is a partial order on fragments. We have:

Lemma 2. *Let t be a synchronizer of a sound FC-CWF net. If \mathcal{W}_t is minimal with respect to the partial order on fragments, then all non-synchronizers of \mathcal{W}_t can be removed by means of applications of the d-shortcut and merge rules.*

Proof Sketch. Intuitively, synchronizers are points where loops begin and end. For two distinct synchronizers of a *minimal* fragment, any occurrence sequence starting from the marking enabling one of them, ending in the marking enabling the other, and in which no other synchronizers occur, is acyclic. Thus we can reduce the possible paths from one synchronizer to another to a single transition using our rules. We do so by constructing auxiliary acyclic workflow nets and reducing those, applying the same reduction rules to our original net.

In our example, the fragment of Figure 4 on the left is reduced to the synchronizer-only fragment shown in Figure 4 on the right. In such a fragment, a marking always marks exactly the places of one of the clusters, and nothing else. Intuitively, the synchronizer-only fragment is an *S-net*, i.e., a net where every transition has exactly one input and one output place, but in which some places are *duplicated*. Figure 3 shows an example of an S-net, while the net on the right of Figure 4 is an S-net in which place c_{10} is duplicated in place c_{11} .

When reducing S-nets we must be careful that the shortcut rule does not “run into cycles”. Consider for instance the second net in Figure 3. If instead of shortcutting t_4 we shortcut t_1 , we obtain a new transition t_7 with i and c_2 as input and output place. If we now shortcut t_7 , we return to the original net with an additional transition connecting i and o . This problem is solved by imposing an (arbitrary) total order on the clusters. Using this order we classify transitions as “forward” (leading to a greater cluster) and “backward” (leading to a smaller cluster). Running into cycles is avoided by only applying the shortcut rule to the backward transition leading to a minimal cluster. Ultimately, this procedure reduces the fragment to an acyclic net. The total number of synchronizers is thus reduced, until none are left. At this point, by Lemma 1 the net is acyclic, and Theorem 2 can be applied. The complete reduction algorithm is listed as Algorithm 1. The algorithm contains several points where the computation might end if some condition is fulfilled. If the net was free choice, we can then conclude that it is unsound.

We have not yet discussed why a fragment could be malformed as mentioned in Line 3 of the algorithm. The proof that every minimal loop has a synchronizer

Algorithm 1 Reduction procedure for cyclic workflow nets \mathcal{W}

```
1: while  $\mathcal{W}$  is cyclic do
2:    $c \leftarrow$  a minimal synchronizer of  $\mathcal{W}$  ▷ If there is none, return
3:    $F \leftarrow$  the fragment of  $c$  ▷ If fragment is malformed, return
4:   while  $F$  contains non-synchronizers do
5:     apply the merge rule exhaustively
6:     apply the iteration rule exhaustively
7:     apply the d-shortcut rule to  $F$  ▷ If not possible, return
8:   end while
9:   fix a total order on  $F$ 
10:  while  $F$  is cyclic do
11:    apply the merge rule exhaustively
12:    apply the iteration rule exhaustively
13:    apply the shortcut rule to the backward transition which ends at a minimal
      cluster
14:  end while
15: end while
16: while  $\mathcal{W}$  is not reduced completely do
17:   apply the merge rule exhaustively
18:   apply the d-shortcut rule to  $F$  ▷ If neither was possible, return
19: end while
```

also shows something more: tokens can only exit a loop at a cluster that contains a synchronizer, and all tokens exit the loop at the same time. Thus when we compute a fragment and find transitions that lead out of the fragment and whose cluster does not contain a synchronizer, or transitions that partially end outside and partially inside the fragment, we can already conclude that the net is unsound. For more information on how to compute fragments, see the next section.

With some analysis on the number of rule application in the acyclic case as well as the S-net case, we can bound the number of rule application to be polynomial:

Theorem 3. *Every sound FC-CWF net can be summarized in at most $\mathcal{O}(|C|^4 \cdot |T|)$ shortcut rule applications and $\mathcal{O}(|C|^4 + |C|^2 \cdot |T|)$ merge rule applications where C is the set of clusters of the net. Any unsound FC-CWF net can be recognized as unsound in the same time.*

4.1 Summarizing the example

We illustrate our algorithm on the example of the insurance claim of Figure 2. To better illustrate our approach, we replace the part between the places c_7 and c_9 by Figure 4.

Our algorithm begins by checking whether \mathcal{W} is cyclic and finds a minimal synchronizer. This could in our example be c_7 , its fragment is exactly the part of the net depicted in Figure 4 on the left. Since the fragment contains non-synchronizers c_{10}, c_{11} , the while loop of Line 4 is entered. The d-shortcut rule

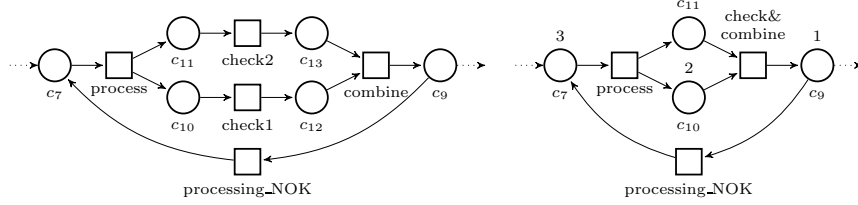


Fig. 4. Extension of the insurance claim net and the synchronizer-only fragment

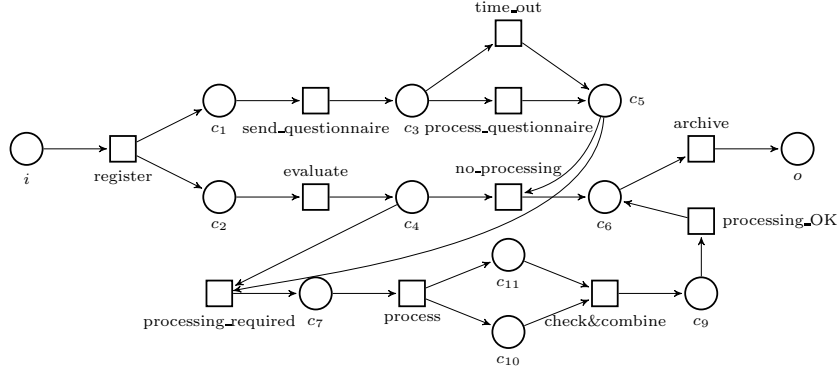


Fig. 5. After shortcutting backward transitions

is applied to **check1** and **check2**. The resulting fragment is depicted in Figure 4 on the right. This fragment consists only of synchronizers and thus the while loop ends. We fix as total order $[c_7] \prec [c_{10}] \prec [c_9]$.

Transition **processing_NOK** is a backward transition as its post-set $[c_7]$ is smaller than its pre-set $[c_9]$ according to the total order. It is shortcut resulting in another backward transition ending in the cluster containing c_{10} , c_{11} , which is then shortcut again to a self-loop on c_9 . The self-loop is removed via the iteration rule.

The resulting net is depicted in Figure 5. This net is acyclic, thus now the d-shortcut and merge rule are applied exhaustively. An intermediate step is depicted in Figure 6. First **process_questionnaire** and **time_out** are merged and the path from i to c_5 is shortcut. Then the linear path from c_7 to o is shortcut into a single transition. Next the path from i to c_4 is shortcut, resulting in the transition **register** to unconditionally enable **no_processing** and **processing_required**. Finally, with three more shortcuts and a merge, the net is completely reduced, and we obtain the transformer shown in Section 2.2.

4.2 Extension to generalized soundness

In [19,1] (see also [16]), two alternative notions of soundness are introduced: k -soundness and generalized soundness. We show that for free choice workflow nets

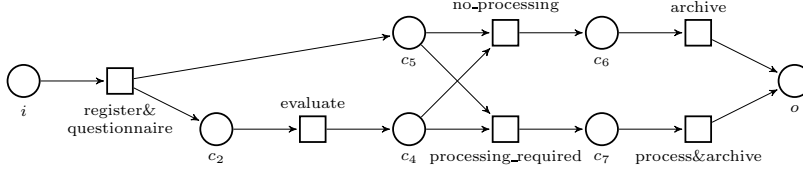


Fig. 6. After some rule applications

they coincide with the standard notion. Therefore, our rules are also complete with respect to these alternative notions.

Definition 15. Let $\mathcal{W} = (P, T, F, i, o)$ be a workflow net. For every $k \geq 1$, let i^k (o^k) denote the marking that puts k tokens on i (on o), and no tokens elsewhere. \mathcal{W} is k -sound if o^k is reachable from every marking reachable from i^k . \mathcal{W} is generalized sound if it is k -sound for every $k \geq 1$.

Theorem 4. Let \mathcal{W} be a free choice workflow net. The following statements are equivalent: (1) \mathcal{W} is sound; (2) \mathcal{W} is k -sound for some $k \geq 1$; (3) \mathcal{W} is generalized sound.

5 Experimental evaluation

We have implemented our reduction algorithm and applied it to a benchmark suite of models previously studied in [17,5].

³ The most complex part of the implementation⁴ is the computation of synchronizers and their fragments. A crucial point is that we are only interested in fragments that consist of free choice places as those are the fragments we might be able to completely reduce. The computation of the synchronizers starts with an overapproximation: starting from a cluster c , we begin by marking for all transitions $t \in c_T$, the places in t^\bullet that are free choice as visited. Whenever we have marked all places in a cluster as visited, we repeat the same for this cluster. In that way we overapproximate the set of clusters that can occur in an occurrence sequence as in the definition of synchronizer. Should all places in c be marked as visited at some point, we consider c a potential synchronizer.

We now compute the fragment of c in a backwards fashion. Starting with only c , we check for every transition whose out-places are contained in the currently identified fragment, whether its in-places were completely marked in the first step. If so, add its in-places and the transition to the fragment. We also check simple soundness properties, e.g. that no transition exists which starts in the fragment and ends partially inside and partially outside the fragment.

³ Nets can be obtained under http://svn.gna.org/viewcvs/*checkout*/service-tech/trunk/_meta/nets/challenge/ in folders `sap-reference` and `ibm-soundness`

⁴ Can be obtained under <https://www7.in.tum.de/tools/workflow/index.php>

	#	P			T			red.	# rule
	nets	avg.	med.	max	avg.	med.	max	by	appl.
Acyclic FC sound	446	20.7	13	154	13.1	9	95	—	12.8
Acyclic FC uns.	761	60.4	49	264	41.1	33	285	73.6%	38.0
Cyclic FC sound	24	46.1	43	118	34.3	26	93	—	43.2
Cyclic FC uns.	155	73.2	61	274	51.1	44	243	78.1%	53.2
Acyclic not FC	542	47.0	38	262	46.8	37	267	68.4%	38.4
Cyclic not FC	30	85.6	72	193	88.1	72	185	66.4%	82.7

Table 2. Analyzed workflow nets

We have conducted some experiments to obtain answers to the following two questions: (1) Since our rules must preserve not only soundness, but also the input/output relation, they cannot be as “aggressive” as previous ones. So it could be the case that they only lead to a small reduction factor in the non-free choice case. To explore this question, we experimentally compute the reduction factor for non-free choice benchmarks. (2) While Theorem 3 is a strong theoretical result (compared to PSPACE-hardness of soundness for arbitrary workflow nets), the $\mathcal{O}(|C|^4 \cdot |T|)$ bound has rather high exponents, and could potentially lead to an impractical reduction algorithm. To explore if the worst case appears in practice, we compute the number of rule applications for free choice benchmarks.

We have used the benchmark suites of [17,5], both consisting of industrial examples. We analyzed a total of 1958 nets, of which 1386 were free choice. Running the reduction procedure for all benchmarks took 6 seconds. The results are shown in Table 2. The number of places and transitions are always given as average/median/max. In the free choice case, our algorithm found that 470 nets were sound (i.e. those nets were reduced completely), and on average the nets were reduced to about 23% of their original size. In the non-free choice case no net could be reduced completely (which does not necessarily mean they are all unsound). However, the size of the nets was still reduced to about 35% of their original size. While we have omitted some more data on the number of rule applications due to lack of space, our experiments indicate that the number of rule applications is close to linear in the size of the net.

6 Conclusion

We have presented the first set of reduction rules for colored workflow nets that preserves not only soundness, but also the input/output relation, and is complete for free choice nets. We have also designed a specific reduction algorithm. Experimental results for 1958 workflow nets derived from industrial business processes show that the nets are reduced to about 30% of their original size.

Our rules can be used to prove properties of the input/output relation by computing it. To reduce the complexity of the computation, we observe that our reduction rules are easily compatible with abstract interpretation techniques:

given an abstract domain of data values, the rules can be adapted so that, instead of computing the transformers of the new transitions using the union, join, and Kleene-star operators, they compute their abstract versions. We plan to study this combination in future research.

Acknowledgements Thank you very much to Karsten Wolf for pointing us to the benchmarks. Many thanks to the anonymous reviewers for the helpful comments.

References

1. Jordi Cortadella and Wolfgang Reisig, editors. *Applications and Theory of Petri Nets 2004, 25th International Conference, ICATPN 2004, Bologna, Italy, June 21-25, 2004, Proceedings*, volume 3099 of *Lecture Notes in Computer Science*. Springer, 2004.
2. Jörg Desel and Javier Esparza. *Free choice Petri nets*, volume 40. Cambridge university press, 2005.
3. Javier Esparza and Jörg Desel. On negotiation as concurrency primitive. In Pedro R. D’Argenio and Hernán C. Melgratti, editors, *CONCUR*, volume 8052 of *Lecture Notes in Computer Science*, pages 440–454. Springer, 2013.
4. Javier Esparza and Jörg Desel. On negotiation as concurrency primitive II: Deterministic cyclic negotiations. In Anca Muscholl, editor, *FoSSaCS*, volume 8412 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2014.
5. Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *Business Process Management*, pages 278–293. Springer, 2009.
6. Cédric Favre, Dirk Fahland, and Hagen Völzer. The relationship between workflow graphs and free-choice workflow nets. *Inf. Syst.*, 47:197–219, 2015.
7. John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
8. Kurt Jensen and Lars M Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.
9. Chun Ouyang, Eric Verbeek, Wil M. P. van der Aalst, Stephan Breutel, Marlon Dumas, and Arthur H. M. ter Hofstede. Wofbpel: A tool for automated analysis of BPEL processes. In Boualem Benatallah, Fabio Casati, and Paolo Traverso, editors, *Service-Oriented Computing - ICSOC 2005, Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005, Proceedings*, volume 3826 of *Lecture Notes in Computer Science*, pages 484–489. Springer, 2005.
10. Shazia Sadiq, Maria Orlowska, Wasim Sadiq, and Cameron Foulger. Data flow and validation in workflow modelling. In *Proceedings of the 15th Australasian database conference-Volume 27*, pages 207–214. Australian Computer Society, Inc., 2004.
11. Wasim Sadiq and Maria E Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2):117–134, 2000.
12. Nikola Trčka, Wil MP Van der Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *Advanced Information Systems Engineering*, pages 425–439. Springer, 2009.
13. Wil Van Der Aalst and Kees Max Van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.

14. Wil M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
15. Wil MP van der Aalst, Alexander Hirnschall, and HMW Verbeek. An alternative way to analyze workflow graphs. In *Advanced Information Systems Engineering*, pages 535–552. Springer, 2002.
16. Wil MP van der Aalst, Kees M van Hee, Arthur HM ter Hofstede, Natalia Sidorova, HMW Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
17. Boudewijn F van Dongen, Monique H Jansen-Vullers, HMW Verbeek, and Wil MP van der Aalst. Verification of the sap reference models using epc reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007.
18. Boudewijn F van Dongen, Wil MP Van der Aalst, and Henricus MW Verbeek. Verification of epcs: Using reduction rules and petri nets. In *Advanced Information Systems Engineering*, pages 372–386. Springer, 2005.
19. Kees M. van Hee, Natalia Sidorova, and Marc Voorhoeve. Generalised soundness of workflow nets is decidable. In Cortadella and Reisig [1], pages 197–215.
20. H. M. W. (Eric) Verbeek, Twan Basten, and Wil M. P. van der Aalst. Diagnosing workflow processes using woflan. *Comput. J.*, 44(4):246–279, 2001.
21. HMW Verbeek, Moe Thandar Wynn, Wil MP van der Aalst, and Arthur HM ter Hofstede. Reduction rules for reset/inhibitor nets. *Journal of Computer and System Sciences*, 76(2):125–143, 2010.

A Appendix

A.1 Proof of Section 3

Theorem 1 *The merge, shortcut and iteration rules are correct for CWF nets.*

Proof. Correctness of the merge and iteration rules follows easily from the definitions. We concentrate on the shortcut rule, whose proof is more delicate.

Assume the transition t and cluster c are as in Definition 10. We say that c occurs in a firing sequence σ if some transition $t \in c \cap T$ occurs in σ .

Let \mathcal{W}_2 be the result of applying the shortcut rule to \mathcal{W}_1 . The proof is divided into four parts.

- (1) For every initial firing sequence σ_2 of \mathcal{W}_2 , there is an initial firing sequence σ_1 of \mathcal{W}_1 leading to the same marking.
Let σ_1 be the result of replacing all occurrences of t'_s (as in Definition 10) in σ_2 by the sequence tt' . Clearly this yields an initial firing sequence σ_1 of \mathcal{W}_1 . The marking reached by these two sequences is the same.
- (2) For each initial firing sequence σ_1 of \mathcal{W}_1 , there is an initial firing sequence σ_2 of \mathcal{W}_1 leading to the same marking.
Since t unconditionally enables c , between any two occurrences of t in σ_1 there is an occurrence of some $t' \in c \cap T$. Define σ_2 as the result of, for each occurrence of t , removing the next occurrence of $t' \in c \cap T$, and replacing t by an occurrence of t'_s . Should the last occurrence of t not be followed by an occurrence of c , we instead start with $\sigma_1 t'$ for some t' . This yields a firing

sequence σ_2 in \mathcal{W}_2 , the corresponding sequence to σ_1 , which leads to the same marking as σ_1 (or $\sigma_1 t'$).

- (3) If \mathcal{W}_1 is sound then \mathcal{W}_2 is sound.

We first show that every transition in \mathcal{W}_2 can be enabled by some initial firing sequence. Since every transition in \mathcal{W}_1 can be enabled by some initial firing sequence, using the corresponding sequence in \mathcal{W}_2 , which exists by (2), we are done for all transitions but those in $c \cap T$. If c still exists in \mathcal{W}_2 , there must be some $t'' \in T$ such that $t'' \neq t$ and $p \in t''^\bullet$ for some $p \in c \cap P$. Since $t'' \neq t$, the transition t'' is unchanged in \mathcal{W}_2 . By soundness of \mathcal{W}_1 , some initial firing sequence σ_1 enables t'' . We extend it by an occurrence of t'' and then to a firing sequence $\sigma_1 t'' \rho_1$ leading to the final marking in \mathcal{W}_1 , which is possible since \mathcal{W}_1 is sound. This sequence contains an occurrence of c which is not matched by a prior occurrence of t , and so does the corresponding sequence in \mathcal{W}_2 . Together with the fact that c is a free choice cluster, it follows that all transitions in $c \cap T$ can be enabled in \mathcal{W}_2 .

We now prove that every initial firing sequence σ_2 in \mathcal{W}_2 can be extended to a sequence that ends with the final marking. Take the corresponding occurrence sequence σ_1 in \mathcal{W}_1 , and extend it to a sequence $\tau_1 = \sigma_1 \rho_1$ that ends with the final marking in \mathcal{W}_1 (possible by soundness of \mathcal{W}_1). The corresponding sequence in \mathcal{W}_2 is $\tau_2 = \sigma_2 \rho_2$, which is the extension of σ_2 (by construction of corresponding sequences) that ends with the final marking.

- (4) If \mathcal{W}_2 is sound then \mathcal{W}_1 is sound.

Since every transition in \mathcal{W}_2 can be enabled by some initial occurrence sequence, using the corresponding sequence in \mathcal{W}_1 we see that the same is true for all transitions but those in $c \cap T$. However, it is then easy to show that those transitions can be enabled in \mathcal{W}_1 : take the initial firing sequence that enables t and extend it by t which unconditionally enables c .

For an initial occurrence sequence σ_1 in \mathcal{W}_1 , the corresponding occurrence sequence σ_2 in \mathcal{W}_2 can be extended to a sequence $\tau_2 = \sigma_2 \rho_2$ that ends with the final marking in \mathcal{W}_2 . The corresponding sequence τ_1 in \mathcal{W}_1 is either $\sigma_1 \rho_1$ or $\sigma_1 t' \rho_1$ for some $t' \in c \cap T$, an extension of σ_1 that ends with the final marking.

A.2 Proof of Section 4

We recall some important results from [2] about free choice nets and S-components. Recall that a net is like a workflow net, but without a distinguished input and output place (see e.g. [2]). A Petri net is a pair (N, M_0) , where N is a net and M_0 is a marking of N called the initial marking.

Definition 16 (S-component). *Let $N = (P, T, F)$ be a net. An S-component of N is a net $N' = (P', T', F')$ such that*

- $\emptyset \neq P' \subseteq P$
- $\bullet s \cup s^\bullet \subseteq T'$ for every $s \in P'$

- N' is a strongly connected S-net (every transition has exactly one pre- and post place in N')

We recall the following theorem:

Theorem 5 (S-component coverability). [2] *A live and bounded free choice Petri net N can be covered by S-components, i.e. there are S-components of N such that every place belongs to some S-component.*

This has implications for sound free choice workflow nets \mathcal{W} :

Definition 17. *For a workflow net \mathcal{W} , let $\overline{\mathcal{W}}$ be the result of adding to \mathcal{W} a transition t^* with o as input and i as output place. We call $\overline{\mathcal{W}}$ the extended net of \mathcal{W} .*

Since \mathcal{W} is sound, the Petri net $(\overline{\mathcal{W}}, i)$ (i.e., the net $\overline{\mathcal{W}}$ with initial marking i) is live and bounded [14] and thus can be covered by S-components.

Theorem 2 *The merge and d-shortcut rule are complete for acyclic FC-CWF nets.*

We split the proof into three lemmas from which the result follows. We call a net *irreducible* if none of our rules is applicable.

Lemma 3. *Let \mathcal{W} be an sound FC-CWF net that is irreducible and let $p \in P$ be a place of \mathcal{W} with $|p^\bullet| > 1$. Then every S-component of $\overline{\mathcal{W}}$ contains an element in $[p] \cap P$.*

Proof. We proceed in two steps.

- (a) There is a transition t in p^\bullet such that: either $t^\bullet = o$ or $[q] \cap P \subseteq t^\bullet$ for some $q \in P$ with $|q^\bullet| > 1$.

This is the core of the proof. We first claim: if $[q] \cap P \subseteq t^\bullet$ for some $t \in [p] \cap T, q \in P$, then (a) holds. Indeed: if $[q] \cap P \subseteq t^\bullet$ for some $t \in [p] \cap T, q \in P$, then either $q = o$ or $|q^\bullet| > 1$, because otherwise the d-shortcut rule can be applied to p, t and q , contradicting the irreducibility of \mathcal{W} . This proves the claim.

It remains to prove that $[q] \cap P \subseteq t^\bullet$ for some $t \in [p] \cap T, q \in P$. For this, we assume the contrary, and prove that \mathcal{W} contains a cycle, contradicting the hypothesis.

Since the merge rule is not applicable to \mathcal{W} , p^\bullet contains two transitions t_1, t_2 such that $t_1^\bullet \neq t_2^\bullet$. We proceed in three steps.

- (a1) For every reachable marking M that marks $[p] \cap P$ there is a sequence σ such that $M \xrightarrow{t_1\sigma} M_1$ and $M \xrightarrow{t_2\sigma} M_2$ for some markings M_1, M_2 , and the sets A_1 and A_2 of clusters marked by M_1, M_2 are disjoint.

Let σ be a longest occurrence sequence such that $M \xrightarrow{t_1\sigma} M_1$ and $M \xrightarrow{t_2\sigma} M_2$ for some markings M_1, M_2 (notice that σ exists, because all occurrence sequences of \mathcal{W} are finite by acyclicity). We have $A_1 \cap A_2 = \emptyset$, because otherwise we can extend σ by firing any transition of any cluster marked by both markings.

- (a2) For every $a_1 \in A_1$ there is a path leading from some $a_2 \in A_2$ to a_1 , and for every $a_2 \in A_2$ there is a path leading from some $a_1 \in A_1$ to a_2 .

By symmetry it suffices to prove the first part. Since A_1 and A_2 are disjoint, a_1 is marked by M_1 but not by M_2 . Thus there is a place q in a_1 that is not marked by M_2 . Since, the sequences $t_1\sigma$ and $t_2\sigma$ only differ in their first element, it must hold that $q \in t_1^\bullet$ and $q \notin t_2^\bullet$. Let S be an S-component of \overline{W} that contains q . Then S contains $^\bullet q$ and thus t_1 . It also contains some place in $a_{1P} = ^\bullet t_1$ and therefore also each transition in a_{1T} and in particular t_2 . In S , let q' be the target of t_2 . It is clear that $q' \neq q$ and even $q' \notin a_{1P}$ as an S-component can only contain one place per cluster.

Comparing the markings after t_1 fired and after t_2 fired, the token of S is in q in the first case and in q' in the second case. This token will in both cases remain there during the sequence σ , thus M_2 marks q' while M_1 marks q . (see Figure 7).

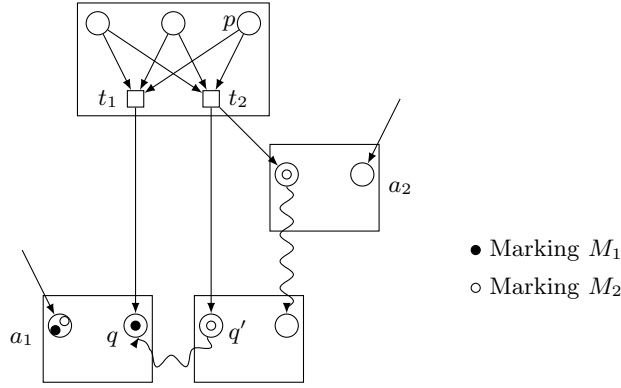


Fig. 7. Illustration of the proof of Lemma 3.

We first show that there is a path from $[q']$ to $[q]$. By assumption, there is no transition of $[p] \cap T$ such that $[q] \cap P \subseteq t^\bullet$ for some $q \in P$, and so $[q] \cap P \not\subseteq t_1^\bullet$. Thus $[q] \cap P$ contains a place $r \neq q$ such that $r \notin t_1^\bullet$, and since M_1 marks $[q]$ (and therefore r), it holds that either some transition in σ marked r or r was already marked by M . Therefore M_2 must also mark r .

Since M_2 marks r , and \mathcal{W} is sound, there is a sequence of transitions τ such that $M_2 \xrightarrow{\tau} M'_2$ and M'_2 marks $[r]$. Since in M_2 , the token in the S-component of q is on q' , there is a path from q' to q and thus from $[q']$ to $[q]$.

We now prove that there is a path from some $a_2 \in A_2$ to $[q']$. If $[q']$ is marked by M_2 , then $[q'] \in A_2$ and we are done. If $[q']$ is not enabled at M_2 (as in the figure) then, since M_2 marks q' and \mathcal{W} is sound, there is a sequence of transitions τ such that $M_2 \xrightarrow{\tau} M'_2$ and M'_2 enables $[q']$. Since A_2 is the set of clusters enabled at M_2 , we have $\tau = t\tau'$ for some transition t with $[t] \in A_2$. So some subword of τ is a path from some cluster of a_2 to $[q']$.

(a3) \mathcal{N} contains a cycle.

Follows immediately from (a2) and the finiteness of N_1 and N_2 .

(b) Every S-component of $\overline{\mathcal{W}}$ contains an element in $[p] \cap P$.

By repeated application of (a) we find a chain of clusters and transitions $(a_1, t_1) \dots (a_k, t_k)$ such that $a_1 = [p]$, $a_k = o$, $t_i \in a_i \cap T$ and $a_{i+1} \cap T \subseteq t_i^\bullet$ for every $1 \leq i \leq k-1$. Since every S-component contains o , every S-component must contain t_{n-1} and also an element of a_{n-1} , and also an element of a_{n-2}, \dots, a_1 .

Lemma 4. *Let \mathcal{W} be a sound acyclic FC WF net that is irreducible. Every S-component contains a place of every cluster, and for every cluster $a \neq [o]$ and every transition t of $a \cap T$ there is a cluster a' satisfying $t^\bullet = a' \cap P$.*

Proof. We first show that every S-component contains a place of every cluster. By Lemma 3, it suffices to prove that every cluster $a \neq [o]$ contains more than one transition. Assume the contrary, i.e., some cluster different from $[o]$ contains only one transition. Since, by soundness, every transition can occur, there is an occurrence sequence $(a_0, t_0)(a_1, t_1) \dots (a_k, t_k)$ such that a_k contains only one transition and all of a_0, \dots, a_{k-1} contain more than one transition. By Lemma 3, every S-component contains a place in all of a_0, a_1, a_{k-1} . It follows that $a_{i+1} \cap T \subseteq t_i^\bullet$ $0 \leq i \leq k-1$. In particular, $a_k \cap T \subseteq t_{k-1}^\bullet$. But then, since a_k only has one transition, the d-shortcut rule can be applied to a_{k-1}, t_{k-1}, a_k , contradicting the hypothesis that \mathcal{W} is irreducible.

For the second part, assume there is a cluster $a \neq [o]$, a transition t of a , and two clusters a_1, a_2 such that $a_1 \cap P \cap t^\bullet \neq \emptyset \neq a_2 \cap P \cap t^\bullet$. Let p_1 be some place in $a_1 \cap P \cap t^\bullet$ and p_2 some place in $a_2 \cap P \cap t^\bullet$. By the first part, every S-component contains a place in a , a_1 and a_2 . Since \mathcal{W} is sound, some reachable marking M marks a . Moreover, since all S-components contain a place in a , and every S-component contains exactly one token, the marking M marks exactly a . Let M' be the marking given by $M \xrightarrow{t} M'$. Since the S-component of p_1 contains a place in every cluster, no cluster different from a_1 can be marked at M' . Symmetrically, no cluster different from a_2 can be enabled at M' . So M' does not mark any cluster, contradicting that \mathcal{W} is sound.

Therefore we have that $t^\bullet \subseteq a' \cap P$ for every cluster $a \neq [o]$ and every transition t of $a \cap T$. To show equality, again assume the contrary. Then with the same reasoning as above, after an occurrence sequence that ends with a , only places in $a \cap T$ are marked but $a \cap T$ is not marked. Thus the marking does not mark any cluster, again contradicting soundness.

Theorem 6. *Let \mathcal{W} be an irreducible sound acyclic FC WF net. Then \mathcal{W} contains only two clusters $[i], [o]$.*

Proof Sketch. Assume \mathcal{W} contains more than two clusters. For every cluster $a \neq [o]$, let $l(a)$ be the length of the longest path from a to o in the graph of \mathcal{W} . Let a_{\min} be any cluster such that $l(a_{\min})$ is minimal, and let t be an arbitrary transition of a_{\min} (notice that a is cannot be $[i]$). By Lemma 4 there is a cluster a' such that $a' \cap T = t^\bullet$. If $a' \neq [o]$ then by acyclicity we have $l(a') < l(a_{\min})$,

contradicting the minimality of a_{\min} . So we have $t^\bullet = [o]$ for every transition t of a_{\min} . If a_{\min} has more than one transition, then the merge rule is applicable. Otherwise, since \mathcal{W} is strongly connected some transition t must have an out-place in $a \cap P$ and thus by 4 $t^\bullet = a \cap P$. The d-shortcut rule is applicable to $[t], t$ and a . In both cases we get a contradiction to irreducibility.

The above lemmas prove that as long as the FC WF net \mathcal{W} consists of more than two clusters and one transition, one of the rules is applicable. We now show that an application of the rules actually summarizes the net in polynomial time.

Definition 18. For every transition t , let $shoc(t)$ be the length of a longest maximal occurrence sequence containing t minus 1, and let $Shoc(\mathcal{W}) = \sum_{t \in T} shoc(t)$.

Notice that if \mathcal{W} has K clusters then $shoc(t) \leq K-1$ holds for every transition t . Further, if $K = 2$ then $Shoc(\mathcal{W}) = 0$.

Theorem 7. Every FC WF net $\mathcal{W} = (P, T, F, i, o)$ can be completely reduced by means of $|T|$ applications of the merge rule and $Shoc(\mathcal{W})$ applications of the d-shortcut rule.

Proof. Let \mathcal{W} and \mathcal{W}' be FC WF nets such that \mathcal{W}' is obtained from \mathcal{W} by means of the merge or the d-shortcut rule. Assume that the merge rule is applied. Then we have $|T'| < |T|$ and $Shoc(\mathcal{W}') \leq Shoc(\mathcal{W})$ because the rule reduces the number of transitions by one.

Now assume that the d-shortcut rule is applied. Let t be the removed transition, c' the cluster that t enabled unconditionally, t' the single transition of c' and t'_s be the new transition produced by the rule. It holds that $|T'| = |T|$ because one transition was removed and one was added. Let σ be any maximal occurrence sequence.

If σ does not contain t , it still exists in \mathcal{W}' . If σ contains t , it also contains t' because t unconditionally enables c' which has only a single transition. In \mathcal{W}' , the corresponding sequence to σ is shorter by one as the occurrences of t and t' have been combined into a single occurrence of t'_s . Thus $shoc(t'_s) < shoc(t)$.

For all other transitions, $shoc(t'')$ either does not change (if one of the longest maximal occurrence sequences does not contain t) or decreases by one (otherwise). Thus in total, $Shoc(\mathcal{W}') < Shoc(\mathcal{W})$.

Lemma 1 Every sound cyclic FC WF net has a synchronizer.

Proof. We begin by showing that every cyclic FC WF net has a loop.

Let π be a cycle of the graph of the net \mathcal{W} . Let t_1 be an arbitrary transition occurring in π , and let t_2 be its successor in π . $t_1^\bullet \neq \{o\}$ because o has no outgoing transitions, and hence no cycle contains o .

By soundness some reachable marking M_1 enables t_1 . Furthermore it holds that $t^\bullet \cap t_2^\bullet \neq \emptyset$. Let M'_1 be the marking reached after firing t_1 from M_1 . Again by soundness, there is an occurrence sequence from M_1 that leads to the final marking. This sequence has to contain an occurrence of a transition of the cluster

$[t_2]$ because there is a token on at least one place of this cluster. In particular, some prefix of this sequence leads to a marking M_2 that enables t_2 .

Repeating this argument arbitrarily for the transitions $t_1, t_2, t_3, \dots, t_k = t_1$ of the cycle π , we conclude that there is an infinite occurrence sequence, containing infinitely many occurrences of transitions of the cycle π . Since the set of reachable markings is finite, this sequence contains a loop.

Let now σ be a minimal loop, i.e. the loop containing the least amount of transitions possible. We show that σ contains a transition belonging to a synchronizer. For this proof, we will need to have a decomposition of \mathcal{W} into S-components which corresponds to the idea of agents in a negotiation. Remember that the number of tokens in an S-component is constant.

Let M be a marking where some transition t_1 in σ is enabled. By soundness, there is some occurrence sequence τ enabled at M which leads to the final marking. Let t be the last transition in τ whose cluster has a transition t^* that is contained in σ . We claim that t^* is a synchronizer of σ .

Assume that this is not the case, i.e. there is a marking M^* that enables t^* (and since the net is free choice, also t) and also marks some other places $p \in (\bigcup_{t' \in \sigma, t' \neq t} \bullet t')$. We first pick a subsequence $\tau' = t_1 t_2 t_3 \dots$ of τ such that $t_1 = t$ and $t_i^* \cap \bullet t_{i+1} \neq \text{emptyset}$. Intuitively, we push one token towards the final marking.

We now construct an occurrence sequence starting from the marking M^* as follows: we start by firing t . Afterwards, whenever a transition in σ is enabled, that transition is fired. If no such transition is enabled, but the next transition in τ' is enabled, that transition is fired. Otherwise, we add a minimal transition sequence that either ends with the final marking or enables some transition in σ or τ' .

First observe that there will always be some place $p \in \bullet t_\sigma$ be marked for some $t_\sigma \in \sigma$. This is because initially one of those places is marked, and whenever a transition whose preset includes such a place is fired, it is a transition in σ . Thus the final marking can never be reached, because that marking only marks o and no other place. Furthermore, since τ' is finite, we can only finitely often add transition sequences that enable a transition in τ' and subsequently fire it. Thus it has to be the case that transitions of σ appear infinitely often. (Because of soundness and because the final marking is not reached, the sequence has to be infinite, otherwise there would be a deadlock.)

However, t cannot be enabled ever again: the tokens of the S-components containing t are somewhere along their path towards o (because they have followed τ'). Furthermore, the sequences we add until some transition in σ is enabled increase the number of S-components whose token is on some place where a transition of σ begins. This also can only happen finitely often.

Thus there must be a loop which uses only a subset of transitions in σ contradicting the minimality of σ .

Lemma 2 *For sound FC WF nets, a fragment of a minimal synchronizer can be reduced to only contain synchronizers by means of the d-shortcut and merge rule.*

Proof. We construct an auxiliary net which will be acyclic.

Let $\mathcal{W} = (P, T, F, i, o)$ be a WF net, t a minimal synchronizer, F_t its fragment S the set of synchronizers in F_t . Every cycle in F_t must contain some transition in S by minimality of t . We now describe our auxiliary net:

We take the fragment of t , add a copy of the places $[t] \cap P$ and also an additional start place i' and end place o' . We then redirect all transitions that put tokens on $[t]$ to instead put them on the copy of $[t] \cap P$. We add a transition that takes a token from i' and places one on each place of $[t] \cap P$. Finally, we remove all synchronizers except those in $[t]$ and instead add a transition for each cluster c from which we removed a synchronizer. The newly added transition has as preset the places $c \cap P$ of the cluster and as postset the new output place o' .

To ensure soundness, we remove all non-reachable clusters. Clusters could be unreachable because they do not appear on a path from $[t]$ to another synchronizer's cluster, but e.g. on a path from another synchronizer's cluster to $[t]$.

This net contains all paths in the fragment leading from $[t]$ to any synchronizer in S . Since all cycles in the fragment contain a transition in S , the net is acyclic. It can therefore be reduced by means of the d-shortcut and merge rules.

We apply the same rules sequence (minus the applications that shortcut over one of our newly created transitions which end in o') to the original negotiation. The result is that the transition t is shortcut to directly enable some $t' \in S$. We repeat this for the other synchronizers in S and thus reduce the fragment to synchronizers only.

Theorem 3 *Every sound FC-CWF net can be summarized in at most $\mathcal{O}(|C|^4 \cdot |T|)$ shortcut rule applications and $\mathcal{O}(|C|^4 + |C|^2 \cdot |T|)$ merge rule applications where C is the set of clusters of the net. Any unsound FC-CWF net can be recognized as unsound in the same time.*

Proof. We again use $Shoc(\mathcal{W})$ as in the proof of Theorem 7.

We start by bounding the time it takes to reduce the minimal fragment with clusters C_F and transitions T_F to a synchronizer only fragment. During this reduction, only the d-shortcut and merge rule are applied. We can thus bound the number of merge rule applications by T_F . Using the auxiliary net construction from 2 we see that for each synchronizer, a net with $\mathcal{O}(C_F)$ clusters and $\mathcal{O}(T_F)$ transitions is reduced. The reduction takes at most $\mathcal{O}(|C_F| \cdot |T_F|)$ applications of the d-shortcut rule and takes place at most once for each cluster in \mathcal{W} .

Next we bound the time to reduce a synchronizer only fragment with clusters C_F . We order the clusters in an arbitrary order c_1, c_2, \dots . Consider the vector $(n(c_1), n(c_2), \dots)$ where $n(c)$ is the number of incoming backward transitions to cluster c . With every application of the shortcut rule as described above, the first non-zero entry of this vector decreases, and all zero entries before the decreasing entry stay zero. Each entry $n(v)$ is bounded (after the merge rule is applied) by the number of clusters. Thus the number of shortcut rule applications is bounded by $|C_F|^2$.

After each application of the shortcut rule to a backwards transition, we apply the merge rule exhaustively. Since the cluster c in the definition of the shortcut

rule has at most C_F outgoing transitions inside the fragment (remember that in a synchronizer only fragment, a transition has exactly a cluster as pre- and post place), at most C_F new edges may result from the shortcut rule application, thus the number of merge rule applications is bounded by $|C_F|^3$.

Finally, applying the reduction procedure for acyclic net \mathcal{W}_A with C_A clusters and T_A transitions takes again $|T_A|$ applications of the merge rule and $Shoc(\mathcal{W}_A)$ applications of the d-shortcut rule by Theorem 7. We can bound $Shoc(\mathcal{W}_A)$ by $\mathcal{O}(|C_A| \cdot |T_A|)$ by the definition of $Shoc$.

While the synchronizer only fragment is reduced, we may sometimes shortcut with a cluster which also has transitions leading outside of the fragment. This results in new transitions being produced that lead outside of the fragment. However, the total number of unique transitions produced in this way over during the whole reduction procedure is bounded by $|C| \cdot |T|$ where C is the set of all clusters of the net: each such transition must originate from some cluster, and since we reduce a synchronizer only fragment, the new transition's post places are exactly the post places of an existing transition. Each of those transitions may be produced up to C times and thus there are at most $|C|^2 \cdot |T|$ merge rule applications.

We bound the number of clusters $|C_F|$ in each fragment and $|C_A|$ in the final acyclic net by $|C|$, the number of clusters in the original net. We further bound the number of transitions $|T_F|$ and $|T_A|$ by $|C| \cdot |T|$ by the above observation.

Summing up, we obtain for the shortcut rule $\mathcal{O}(|C| \cdot (|C|^2 + |C|^2 \cdot |T|) + |C|^2 \cdot |T|) = \mathcal{O}(|C|^3 \cdot |T|)$ shortcut rule applications and $\mathcal{O}(|C| \cdot (|C|^3 + |C| \cdot |T|) + |C| \cdot |T| + |C|^2 \cdot |T|) = \mathcal{O}(|C|^4 + |C|^2 \cdot |T|)$ merge rule applications suffice.

A.3 Extension to generalized soundness

We recall some definitions and theorems from [2]

Definition 19 (Trap). A set R of places of a Petri net is a trap if $R^\bullet \subseteq {}^\bullet R$

Definition 20 (Home marking). Let N be a Petri net, M_0 an initial marking. A marking M is a home marking of (N, M_0) if it is reachable from every marking reachable from M_0 .

Theorem 4 Let \mathcal{W} be a free choice workflow net. The following statements are equivalent:

- (1) \mathcal{W} is sound.
- (2) \mathcal{W} is k -sound for some $k \geq 1$.
- (3) \mathcal{W} is generalized sound.

Proof. (1) \Rightarrow (2). Assume \mathcal{W} is sound. Fix $k \geq 1$, and let $i^k \xrightarrow{\sigma} M$ be an arbitrary occurrence sequence of \mathcal{W} . We prove that there exists an occurrence sequence τ such that $M \xrightarrow{\tau} o^k$.

Consider the Petri net \mathcal{N} obtained by adding to \mathcal{W} a transition t^* with o as input and i as output place. Since \mathcal{W} is sound, the Petri net (\mathcal{N}, i) (i.e., the net \mathcal{N}

with initial marking i) is live and bounded [14]. Moreover, the marking i marks all proper traps of \mathcal{N} : indeed, by soundness there is an occurrence sequence $i \xrightarrow{\sigma} i$ such that σ contains all transitions of \mathcal{N} ; since after such a sequence all traps are necessarily marked, the claim is proved.

We now show that the net (\mathcal{N}, i^k) as initial marking also satisfies these two properties. First, by Theorem 4.21, Theorem 4.27, and Theorem 5.8 of [2], adding tokens to a live and bounded marking of a free choice net preserves liveness and boundedness (see also exercise 4.8 of [2]). So (\mathcal{N}, i^k) is also live and bounded. Second, if i marks all proper traps of \mathcal{N} , then obviously so does i^k .

By the Home Marking Theorem ([2], Theorem 8.11), the marking i^k is a home marking of \mathcal{N} . Therefore, there exists an occurrence sequence τ' such that $M \xrightarrow{\tau'} i^k$. Assume that $M(i) = k'$ for some $0 \leq k' \leq k$. Let τ be the result of removing from τ' the last $k - k'$ occurrences of transition t^* . We have $M \xrightarrow{\tau} o^k$, and we are done.

(2) \Rightarrow (1). Assume W is k -sound for some $k \geq 1$. By the definition of k -soundness, the net (\mathcal{N}, i^k) is bounded and deadlock-free. By Theorem 4.31 of [2], (\mathcal{N}, i^k) is live. By Commoner's theorem, (\mathcal{N}, i) is also live, and by Theorem 5.8 of [2] it is also bounded. Therefore, the workflow net W is sound (see [14]).

(3) \Rightarrow (2). Obvious from the definition.

(2) \Rightarrow (3). In the statement (1) \Rightarrow (2) we have actually shown that if W is sound, then W is sound for every $k \geq 1$. So assume that W is k -sound for some $k \geq 1$. Then W is sound, and therefore W is k -sound for every $k \geq 1$.